

A STUDY AND IMPLEMENTATION OF ENCRYPTION, WITH EMPHASIS ON CHAOTIC MAPS

By

TEBOHO M. LEEPILE

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

© Copyright 2006
by
Teboho M. Leepile, 2006

CERTIFICATION OF APPROVAL

A STUDY AND IMPLEMENTATION OF ENCRYPTION, WITH EMPHASIS ON CHAOTIC MAPS

by

Teboho M. Leepile

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:



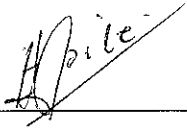
Dr. Varun Jeoti
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

A handwritten signature in black ink, appearing to read 'Teboho M. Leepile', is written over a horizontal line.

Teboho M. Leepile

ABSTRACT

The security of data transmitted over public communication networks and valuable data storage have necessitated the need for very secure cryptography. Applications like video conferencing, cable TV broadcast, etc use encryption extensively. Hence researches for better ways of protecting data are still underway. And this project was aimed at finding secure cipher by implementing Logistic Map Cipher for plaintext encryption and decryption. The research was based on both symmetric ciphers and asymmetric ciphers. The symmetric cryptosystem was chosen and finally implemented. In trying to implement logistic map, Chaotic Maps were briefly analyzed and other types of encryption were investigated in order to understand intensive and extensive applications of cryptography.

The three main parts of focus are keys' generator, the encryption and decryption parts, which are the main steps before cryptanalysis can be carried out. In encryption, the individual success of different parts will guarantee a complete cipher.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr Varun Jeoti for the guidance and assistance he offered me during this project. He helped me put in perspective, the concepts for this project. And thanks to Mr. Mohammad Asim for the help and valuable discussions on Logistic Maps. Without their help the project would not have been successful.

Great deal of appreciations is also extended to the technicians in the Electrical Electronics Department in University Technology Petronas for their help with material and equipment needed for this project, especially Ms Siti Hawa. Their contribution helped a lot in trying to achieve the objectives of this project.

Not forgetting all my classmates and friends for their resourcefulness and willingness to engage in the discussions about this project and for all the assistance they gave me. With your support I kept trying to make it and I thank you all for that.

LIST OF FIGURES

Figure 2.1: Example to symmetric cipher based on Tiny Encryption Algorithm	9
Figure 2.2: Indicates the test result for a block cipher with padding.	10
Figure 2.3: Block chaining mode encryption (adapted from [21])	13
Figure 2.4: Cipher Feedback mode of encryption (adapted from [21])	14
Figure 2.5: Bifurcation diagram for logistic map	17
Figure 2.6: Block diagram for symmetric cipher	18
Figure 3.1: Encryption and decryption cryptosystem	19
Figure 3.2: Project Flowchart	20
Figure 4.1: General block diagram for chaotic cipher internal mechanisms.	24
Figure 4.2: Encryption algorithm for logistic map	25
Figure 4.3: Plaintext and Ciphertext in decimals and ASCII characters	27
Figure 4.4: Iterated values for the logistic equation encrypting 40 characters	28

LIST OF TABLES

Table 1.1: The Comparison of keys performance over the same cryptanalysis 6

Table 3.1: Lists of the equipment necessary for this project . . . 21

Table A: Logistic Equation Parameters 35

KEYWORDS/DEFINITION

No.	Acronym/Term	Definition
1.	RSA	Rivest-Shamir-Adleman
2.	DES	Data Encryption Standard
3.	Plaintext	Data that has not been encrypted or has been decrypted
4.	Ciphertext	The data that has been encrypted
5.	Private Key	It is the key only known by the person who decrypts ciphertext and is used to decrypt.
6.	Public Key	It is the key known in the public channels and is used to encrypt plaintext
7.	PKI	Public Key Infrastructure
8.	AES	Advanced Encryption Standard
9.	Asymmetric Encryption	Cipher that uses two different keys; one key for encryption the other one for decryption.
10.	Symmetric Encryption	Cipher that uses one key for both encryption and decryption
11.	Cipher/Cryptosystem	Algorithm used for encryption and decryption of text, voice, still pictures and video.
12.	3-DES	Triple DES
13.	CA	Certificate Authority
14.	LAN	Local Area Network
15.	IP	Internet protocol

TABLE OF CONTENTS

LIST OF FIGURES	vi
LIST OF TABLES	vii
KEYWORDS/DEFINITIONS	viii
CHAPTER 1:INTRODUCTION	1
1.1 Background of Study	1
1.1.1 Description of Encryption	1
1.2 Problem Statement	2
1.2.1 Scope of Work	2
1.2.2 Objectives	2
1.2.3 Relevance of the Project	3
1.2.4 Feasibility of the Project	3
1.2.5 Organisation of the Report	3
CHAPTER 2:THEORY & LITERATURE REVIEW	5
2.1 Asymmetric Ciphers, PKI	5
2.1.1 Types of Algorithms	6
2.2 Mathematical Modelling for Asymmetric Cipher.	7
2.2.1 Public Key Encryption	7
2.2.2 Private Key Decryption	7
2.2.3 Factoring Part	7
2.3 Symmetric Ciphers	9
2.3.1 Single Bit Cipher	10
2.3.2 Block Cipher	10
2.3.3 Types of Algorithms	11

2.3.4 Modes of Operation	11
2.3.5 Padding Types for Different Modes	14
2.4 Mathematical Modelling for Logistic Map	16
2.4.1 Types of Chaotic Maps	16
2.4.2 Logistic Map Equation	16
2.4.3 The Logistic Map Cipher	17
2.4.4 Security of the Cipher	18
CHAPTER 3:METHODOLOGY/PROJECT WORK	19
3.1 Procedure Identification	19
3.2 Tools Required	21
3.2.1 Software Based Cipher	21
3.2.2 Public Network/Local Area Network	22
CHAPTER 4:RESULTS AND DISCUSSIONS	23
4.1 Results	23
4.1.1 Implementation of Chaotic Cipher	23
4.1.2 Logistic Map Ciphers Implementation	25
4.1.3 Analysis	27
4.1.4 Limitations	29
4.2 Discussions	30
CHAPTER 5:CONCLUSIONS AND RECOMMENDATIONS	31
5.1 Conclusions	31
5.2 Recommendations.	31
REFERENCES	32

APPENDICES	34
Appendix –A: Tools and Materials Required	35
Appendix –B: Source Codes Iterations Projection (MATLAB)	36
Appendix –C: Source Codes Bifurcation Diagram (MATLAB)	37
Appendix –D: Implemented Source Codes Logistic Map Cipher (MATLAB)	39
Appendix –E: Representation of ASCII Characters	41
Appendix –F: Source Codes for Symmetric Cipher with Padding [4] (Java)	43
Appendix –G: Project Gantt chart	45

CHAPTER 1

INTRODUCTION

The computer networks, internet, telephones networks, massive data storage disks and cyber-identity authentication have necessitated software and hardware to protect data transmitted or stored over these media [7]. The security of information is implemented with methods such as encryption, firewalls, smartcards, etc. In this chapter encryption is introduced and briefly discussed at different levels of classification, design, application and advantage of use. It concludes by stating the objective and the scope of the work required in this project.

1.1 BACKGROUND OF STUDY

1.1.1 Description of Encryption

The art of keeping the data transmitted over communication networks or stored in hard-disks safe is accomplished by encryption. Encryption ensures that data kept private and confidential is only accessed by a person with the proper key/password and cryptographic software or hardware [10], [13].

The software or hardware and the key/s used to encrypt and decrypt are called cryptosystem or cipher. The key can be a hexadecimal number, or a password of normal characters of the keyboard which is eventually converted into a hexadecimal number. The cipher is only safe if it has not been broken into. The process of trying to hack into an encryption system is called cryptanalysis [12]. There are hash functions which are classified as part of encryption. The hash function is used to authenticate an encrypted message or for cyber-identity [3], [17].

1.2 PROBLEM STATEMENT

The challenge is to study the methods of designing a cryptographic system using the Chaotic Maps. The Logistic Map is to be used to develop the symmetric cipher for plaintext. The system is required to be easy and cheap to implement on software level. The systems investigated should be able to encrypt and decrypt text message using secret key.

1.2.1 Scope of Work

- The project involves modelling all the necessary concepts of Logistic Maps in mathematical formulae and equations.
- The scope is limited to the encryption of any data represented in digital form without considering the medium of data transmission if data is transmitted over the public networks.
- Delve in the possibility of writing JavaScript algorithm, Java, MATLAB or C++ algorithms to represent the mathematical equations and formulae and create chaos based cipher.

1.2.2 Objectives

- It was to study how symmetric cryptosystem based on Chaotic Maps work.
- Study the steps required to implement the system on simulation software and find software advantages over firmware/hardware.
- Look into mechanism used to develop other standardised and popular cryptographic algorithms.
- Finally develop a symmetric cipher based on Logistic Map

1.2.3 Relevance of the Project

The project is relevant because it studied the use of encryption over the existing public communication networks for data security. If the cryptosystem is designed based on chaotic maps, it would be cheaper and widely accessible because it would not require complicated computations for small microprocessors' applications. It would be relatively fast to compete against other encryption methods because of the simple equations [18]. Thus far the problem is that cryptosystems are expensively implemented and not widely available in gadgets like cell-phones and palmtops.

1.2.4 Feasibility of the Project

The literature review in this report indicates that implementing a cryptosystem using Logistic Maps is possible. It is more easier to implement logistic map on the software because the computer processors no longer have limitations of handling lot of numbers with many digits after the decimals point. Hardware processors still have some computational limitations to perform iterations for decimal points and the processors that can handle floating numbers are expensive [19].

1.2.5 Organisation of the Report

The report starts with the introduction of encryption concepts and classification of the encryption algorithm with their related features. It also briefly touches on Chaotic Maps as methods of encryption for both symmetric and asymmetric cipher. The concepts introduced in literature review are fundamental to encryption and are of interest. The basis of these concepts is to complete all the tasks stated in problem statement, objectives, and scope of work.

The subsequent chapters delve on encryption theory and the concepts that are relevant to the success of encryption theory, like standards, and standardised

encryption algorithms. The methodology in Chapter 3 discusses possible steps to create a working algorithm based on Logistic Maps. It discusses the kind of software that would be necessary for successful implementation.

Finally the results, discussions and conclusions will be stated and elaborated in Chapters 4 and 5 after studying the encryption results of some of the encryption algorithms used for text encryption.

CHAPTER 2

LITERATURE REVIEW AND THEORY

The theory provided in this part of the report explains difference between Public Key Infrastructure (PKI) and Symmetric Cipher. The standardized encryption methods are used to elaborate how different ciphers work. The practical application areas of interested, such as email, smartcards demonstrate extensive need of encryption.

2.1 ASYMMETRIC CIPHERS, PKI

The asymmetric ciphers use the public communication infrastructure to send data between the sender and the receiver. The PKI [5], [14] uses private key and public key for encryption and decryption respectively. The sender can create digital certificates for documents before they can be sent over the public network.

The public key is known by both the sender and the receiver, while the private key is only known by the receiver for decryption of encrypted data. Knowledge of the public key does not let the intruder guess the private key even when using brute force attack [12]. The keys are mostly 128 bits long which so far have proven to be very difficult to crack in most asymmetric cryptosystems.

Keys longer than 100 bits have billions of digits combinations that can take any computer with fast processing speed years to factor all the possible combinations. Table 1.1 compares how many bits are required to encrypt data in RSA (PKI) and symmetric ciphers and the total time it will take to compute the private/secret key fraudulently. It also compares the amount of time and resources required to factor the combinations for a private key.

Table 1.1: Comparison of keys performance over the same cryptanalysis [1]

Symmetric Key Size (Bits)	RSA Key (Size in Bits)	Time to Break	Number of machines	Amount of memory
56	430	< 5 minutes	105	Trivial
80	760	600 months	4,300	4 GB
96	1020	3 million years	114	170GB
128	1620	10^{16} years	16	120TB

2.1.1 Types of Algorithms

The algorithms for PKI are slower than the ones for symmetric ciphers hence they are mainly used for encryption of small amounts of data like in text messages. The PKI system does not work well if the voice or video encryption works in real time, like in audio streaming. In real time streaming of encrypted data, encryption time slows down the data transmission speeds. Rivest Shamir and Adleman (RSA) , Chaotic Maps, Deffie –Hellman (DH) , Eleptic Curve (EC) are some of the standard algorithms used to implement the PKI. However the use of Chaotic Maps is not only limited to the asymmetric cipher because Chaotic Maps are also used to design the symmetric ciphers.

2.2 MATHEMATICAL MODELLING FOR ASYMMETRIC CIPHER

This cryptosystem is broken down into three main parts, encryption, decryption and factoring part and the extra part which is digital signature [8]. The encryption part mixes the public key with the data and enable the person encrypt data. The decryption part decrypts the ciphertext into plaintext using the private key.

2.2.1 Public Key Encryption

Mathematical formula, $[K = B^a \text{ MOD } p]$ is used to generate both public and private keys. A computer algorithm can be developed to implement key generator for the cryptosystem. The computer program that implements encryption of data is mathematically represented as $[C = P^e \text{ MOD } n]$. This formula is used to convert normal plaintext into ciphertext. K which is the key can be used as superscript, e or d depending on whether it encrypts or decrypts. When $K = e$ it encrypts and used to decrypt when $K = d$.

2.2.2 Private Key Decryption

The private key is used to convert the ciphertext back into the original plaintext. The equation, $[P = C^d \text{ MOD } n]$ helps in changing encrypted data into plaintext. The superscript d , is the private key. The computer program can be written to implement these mathematical equations.

2.2.3 Factoring Part

The PKI can only be cracked if small prime numbers are used as the private keys. It has been found that factoring prime numbers is easy to do if they are small. If the value of the private key is very large (*more than 100 bits*) it can not be easily guessed nor generated more than once using the randomizing system. The value of

$\varphi(n) = (p - 1)(q - 1)$ for large values of n is difficult to factor into p and q . The value of $\varphi(n)$ can not be factored without prior knowledge of the factors n , p and q . The value of n is calculated as $n = p*q$. Equations (1.1) and (1.2) are used to calculate p and q values.

$$p - q = [(n - \varphi(n) + 1)^2 - 4*n]^{1/2} \quad (1.1)$$

$$p + q = n - \varphi(n) + 1 \quad (1.2)$$

The challenge is to write programming code for all the mathematical equations as to implement the cryptosystem. There are standard algorithms that have already been written to implement these asymmetric ciphers [4], but one has to learn how these algorithms are implemented to create cryptosystems. The experience gained with working with standardised algorithms can be used to implement Chaotic Map algorithms.

2.3 SYMMETRIC CIPHERS

The symmetric ciphers are mostly used where faster rates of encryption/decryption of large volumes of data are required. The symmetric cipher is different from the PKI cryptosystem because it uses one key for encryption and decryption. The sender and receiver are the only parties that know the key. The users have to find a means of exchanging the key securely so that it does not end up in the hands of the wrong person. Figure 2.1 shows an example of symmetric cipher that uses a single key to encrypt and decrypt.

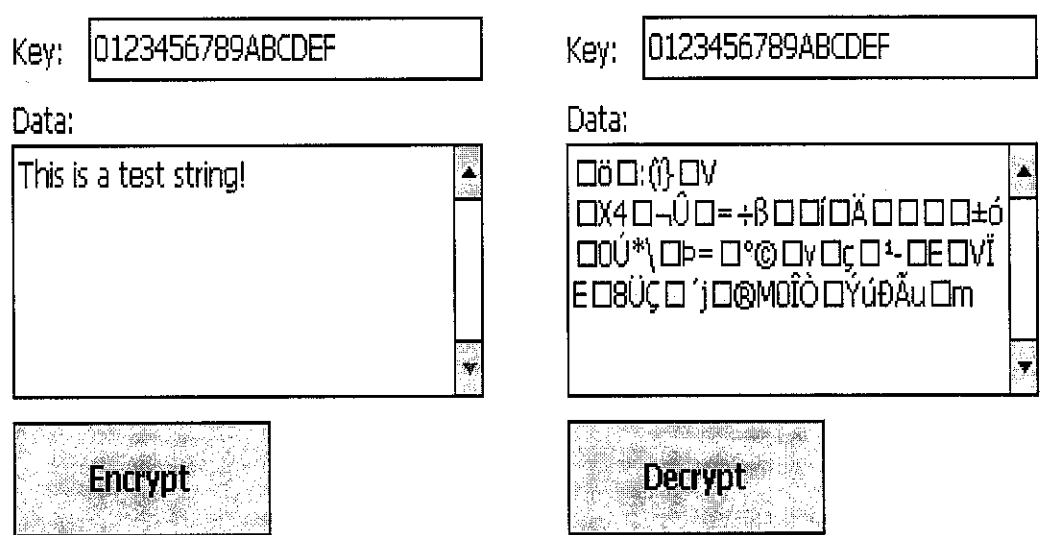


Figure 2.1: Example to symmetric cipher based on Tiny Encryption Algorithm [22].

In order to prevent the intruder getting access to the symmetric key, session keys are used instead of permanent keys. Session keys are symmetric keys that are developed every time there is a need to encrypt data. The cipher is created such that it generates its own keys instead of buying the key from the third party. If the cipher does not allow the user to choose his/her own key, the user can purchase the key from the developer of the cipher, who is referred to as the third party. Third party keys can be compromised by dishonest developers of digital certificates [3].

2.3.1 Single Bit Cipher

The symmetric cipher can also be used to create digital signatures for the data to be transmitted over the network. The different algorithms can be designed so that they encrypt a block of bits instead of single bit at a time. Encrypting single bits is faster and is less prone to errors incase the bits are destroyed while in transit. But incase the intruder intercepts encrypted data bits over the communication networks; he/she can safely add his/her own bits to the encrypted bits if the ciphertext was not signed digitally [8]. RC4 is an example of faster single-bit cipher whose algorithm is easy to implement.

2.3.2 Block Cipher

The block ciphers are slower compared to single bit ciphers. The intruder can insert or delete any bits if they are encrypted as a block. Block ciphers cause more traffic in the network because retransmission of the block of bits puts more bits in the public network, this may slow down the network. If a whole block is deleted by the intruder a large portion of the decrypted bits would not convey the intended message. Figure 2.2 below indicates the results of the sample block cipher algorithm with padding [6].

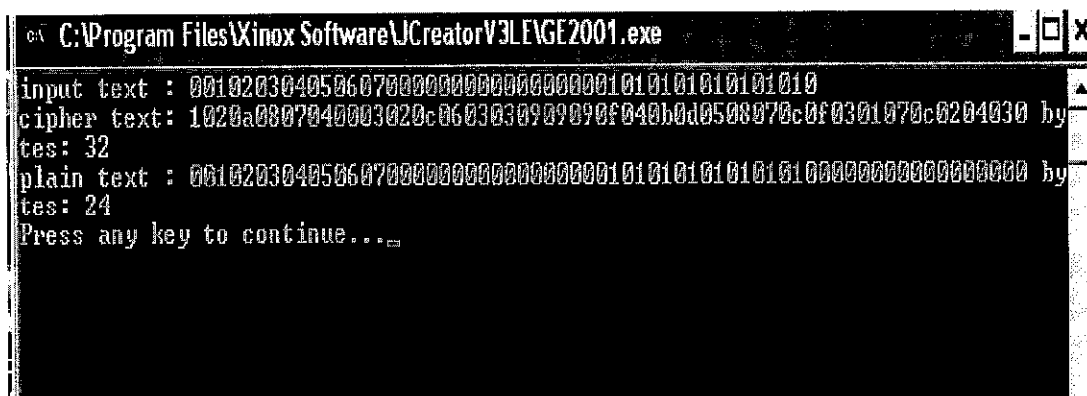


Figure 2.2: Indicates the test result for block cipher with padding (Appendix F)

Padding in block cipher is the method of adding extra bits to the cipher text for additional security. For example, a block cipher could be executing bits of plaintext in multiples of 8 or 16 bytes in length. In case there are less than 8 bytes or 16 bytes to encrypt the cipher will pad the data by adding a block of zeros to the bits of plaintext so that the block has 8 bytes or 16 bytes. The block ciphers execute blocks of predefined bits only. DES and AES are examples of block ciphers.

2.3.3 Types of Algorithms

There are many different types of symmetric ciphers such as EL Gamal, DES, Triple DES, AES, Blowfish, and RC5. These four types of algorithms are still used to develop commercial ciphers. The difference among them is the internal mechanism of producing ciphertext although they are single key based.

2.3.4 Modes of Operation

The way the symmetric ciphers are implemented is called modes. The modes are different ways of making the same algorithm secure. The different modes can offer protection of data against errors and eavesdroppers. They are also used to bring flexibility in the way symmetric cryptosystem is implemented. These Modes [2] have been standardised to ensure that their application is uniform to avoid irregularities.

- **Electronic Code Book (ECB)**

The ECB implements the block cipher and if errors occur to one of the bits of the encrypted data, the whole bit block is erroneous. This mode suffers from bit block insertion by the intruders. The data to be encrypted p is divided into blocks of m bits:

$$P = m_1, m_2, \dots, m_k$$

The cipher blocks are divided into c_1, \dots, c_k and are combined with the key so that they can be defined as follows:

$$c_i = e_k(m_i)$$

The e_k is the key function that produces c_i , which is the encrypted data. The decrypting part undoes what the encryption does.

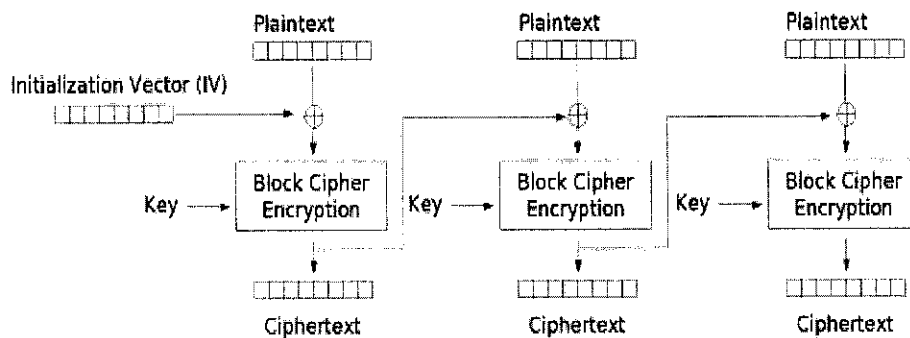
- **Cipher Block Chaining (CBC)**

This cipher system divides data into blocks of bits and chains the encrypted data blocks together to avoid eavesdroppers from inserting their own blocks of bits between the blocks of encrypted information. The equation for encryption is given as follows:

$$c_1 = e_k(m_1 \text{ XOR } IV)$$

$$c_i = e_k(m_i \text{ XOR } c_{i-1}) \text{ for } i > 1$$

The additional value of IV that is passed to all the data to be encrypted (plaintext) ensures that ciphertext blocks look different although they are encrypted with the same secret key. This implies that a different ciphertext is produced every time the same plaintext is encrypted with the same key. An error in one of the blocks of bits results in error in the blocks that follows after that. Figure 2.3 depicts CBC mode of encryption.



Cipher Block Chaining (CBC) mode encryption

Figure 2.3: Cipher Block Chaining mode encryption (Adapted from [21])

- **Output Feedback (OFB)**

This mode converts block cipher into stream cipher. One bit error in the block of ciphertext results into one bit error in decrypted data. The mode is faster because it uses the stream cipher instead of the block cipher.

- **Cipher Feedback Mode (CFB)**

This method also uses block cipher to produce stream cipher. The single bit error in one block affects the same block and the following block of ciphertext. This method is prone to errors compared to the other methods. The key generated in this method is used iteratively and produce different blocks of cipher. Figure 2.4 indicates how CFB is implemented

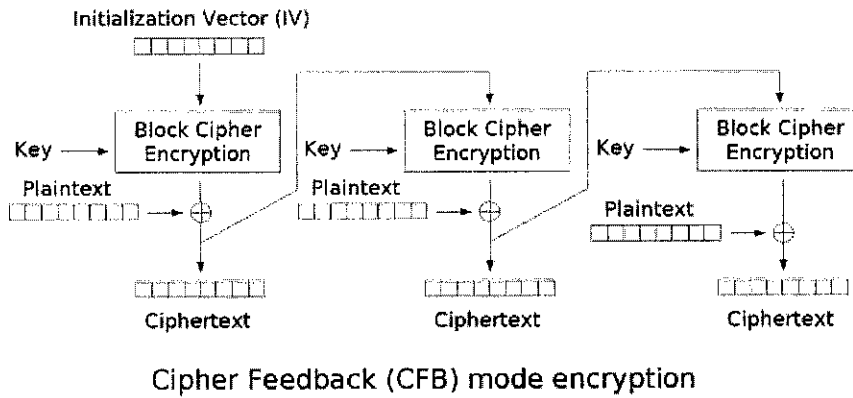


Figure 2.4: Cipher Feedback mode of encryption (Adapted from [21])

Based on the discussion of all the possible algorithm modes, they have their own advantages, but CBC looks more suitable compared to the rest because it prevents insertion or deletion of blocks of bits once data is encrypted.

2.3.5 Padding Types for Different Modes

Since the CBC mode is the preferred mode, its authentication capabilities will be discussed. The CBC data is padded to form a series of n bits block. The padding is addition of bits to a block of bits to be encrypted [4]. The blocks are converted to cipher as explained in CBC mode. Types of padding methods that the project will look into implementing are:

- **Type 1:**

It is the algorithm to add as many zeros as possible to make a whole number of bits in the block of plaintext. It appends zeros if the block has fewer bits than the fixed set of bits that the cipher is configured to execute. Figure 2.2 shows this type of padding. The advantage of this method is that it prevents the CBC mode bits insertion or deletion.

- **Type 2:**

The method adds single 1 bit and a string of zeros to make a whole number of bits in a block of data to be encrypted. This method adds extra bits to the ones that are about to be encrypted. Extra bits are only added if the block of plaintext is small than the number of bits the cipher is specified to encrypt. Block ciphers are specified to encrypt a certain number of bits. For example, DES is specified to encrypt 64 bits of plaintext. If there are less than 64 bits, the 1s and 0s are appended to the plaintext block. The extra bits indicate end of message in case the cipher block ends with a string of zeros.

2.4 MATHEMATICAL MODELING FOR LOGISTIC MAP

2.4.1 Types of Chaotic Maps

The chaos theory stems from mathematical equations that produce random numbers if the initial condition is within a certain range and is iterated for a number of times [11]. The chaotic equations are deterministic in nature, meaning they saturate at a single value after performing few number of iterations. The Chebyshev Map, Tent Map, Relational Map and Logistic Map are few of many chaotic maps that can be used to create ciphers [12], [19]. The commonality among chaotic maps is that under certain, controlled conditions their equations become chaotic.

2.4.2 Logistic Map Equation

The chaos equation that the project looks into is the logistic map, $x_{n+1} = a * x_n(1 - x_n)$ Eq.(2.1). The equation becomes chaotic for x interval of $(0, 1)$ and a interval $[3.5, 4)$ [16]. For any a value less than 3, Eq. (2.1) converges to one particular number and chaos is not attained. And also for x value greater than one, logistic equation grows without bound. The bifurcation diagram for the logistic map in figure 2.5 indicates the area suitable for x and a intervals for Eq. (2.1). On the diagram in figure 2.5, $a = \lambda$ and $y = x_{n+1}$.

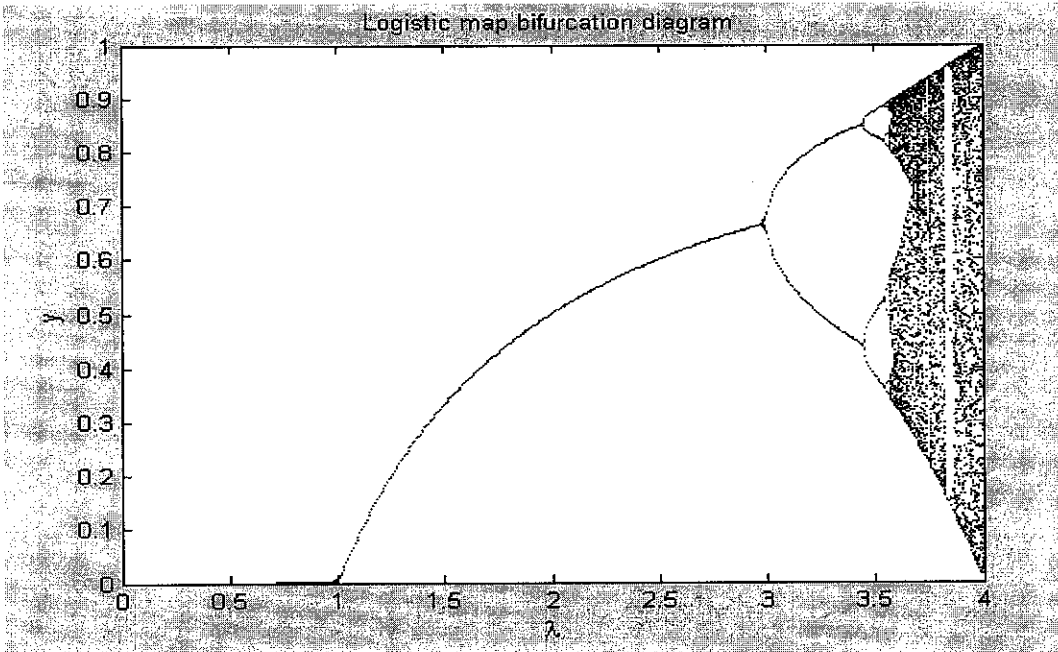


Figure 2.5: Bifurcation diagram for logistic map (Based on appendix C)

The highly shaded area indicates the numbers around which the equation becomes chaotic. Far to the left of the graph the logistic equation is not chaotic.

2.4.3 The Logistic Map Cipher

The cipher is symmetric and has only one key, which is randomly chosen by the user. The x value for Eq. (2.1) is the key for this kind of ciphers. Figure 2.6 indicates the block diagram for the cipher. Plaintext is converted into ciphertext with the use of the secret key.

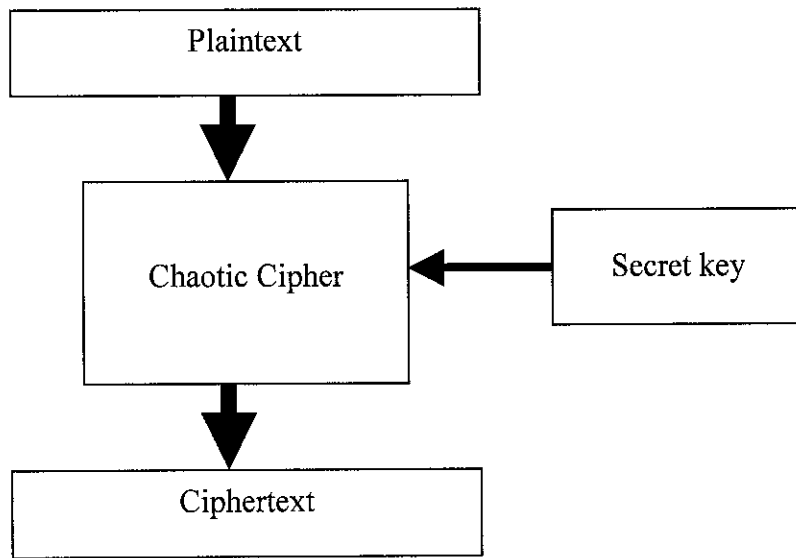


Figure 2.6: Block diagram for symmetric cipher

The encryption block entails the following steps:

1. Choosing the Secret Key (x value)
2. Deciding the chaos constant, a for logistic equation.
3. Iterations of x , to produce the ciphertext.

The decrypting part of the cipher must have the same parameters as the encryption part. For mass distribution of the cipher, steps 2 and 3 above must be the same for all the ciphers, part one should be different for different users, since it is key.

2.4.4 Security of the Cipher

The security of the cipher can be enhanced by using session keys. The session keys are secret keys used for short duration, like few minutes, hours or weeks. This makes the intruder's job difficult because by the time they discover the secret key it is no longer in use.

CHAPTER 3

METHODOLOGY/PROJECT WORK

3.1 PROCEDURE IDENTIFICATION

The cryptosystem is divided into three distinct subsystems. Private and public key generation, encrypting subsystem and decryption subsystem make up the whole crypto system as shown in figure 3.1. The plaintext is mixed with public key and the cipher-text is generated. When the ciphertext reaches the destination it is decrypted back plaintext. The assumption is that the encrypted data is transmitted over the public networks such as internet, GSM communications, etc.

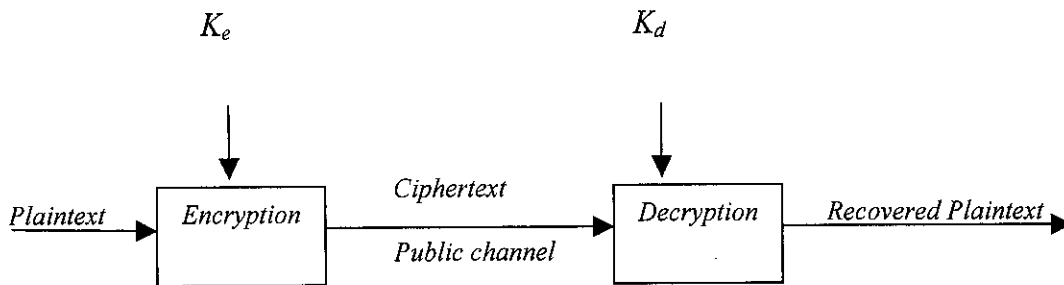


Figure 3.1: Encryption and decryption cryptosystem

K_e is the public key and K_d is the decryption key. For a system created such that $K_e = K_d$, the cipher is symmetric and this option requires that the key be kept secret. The decryption system is the inverse of the encrypting part. The flow chart in figure 3.2 depicts all the steps necessary for software implementation either symmetric or asymmetric cipher.

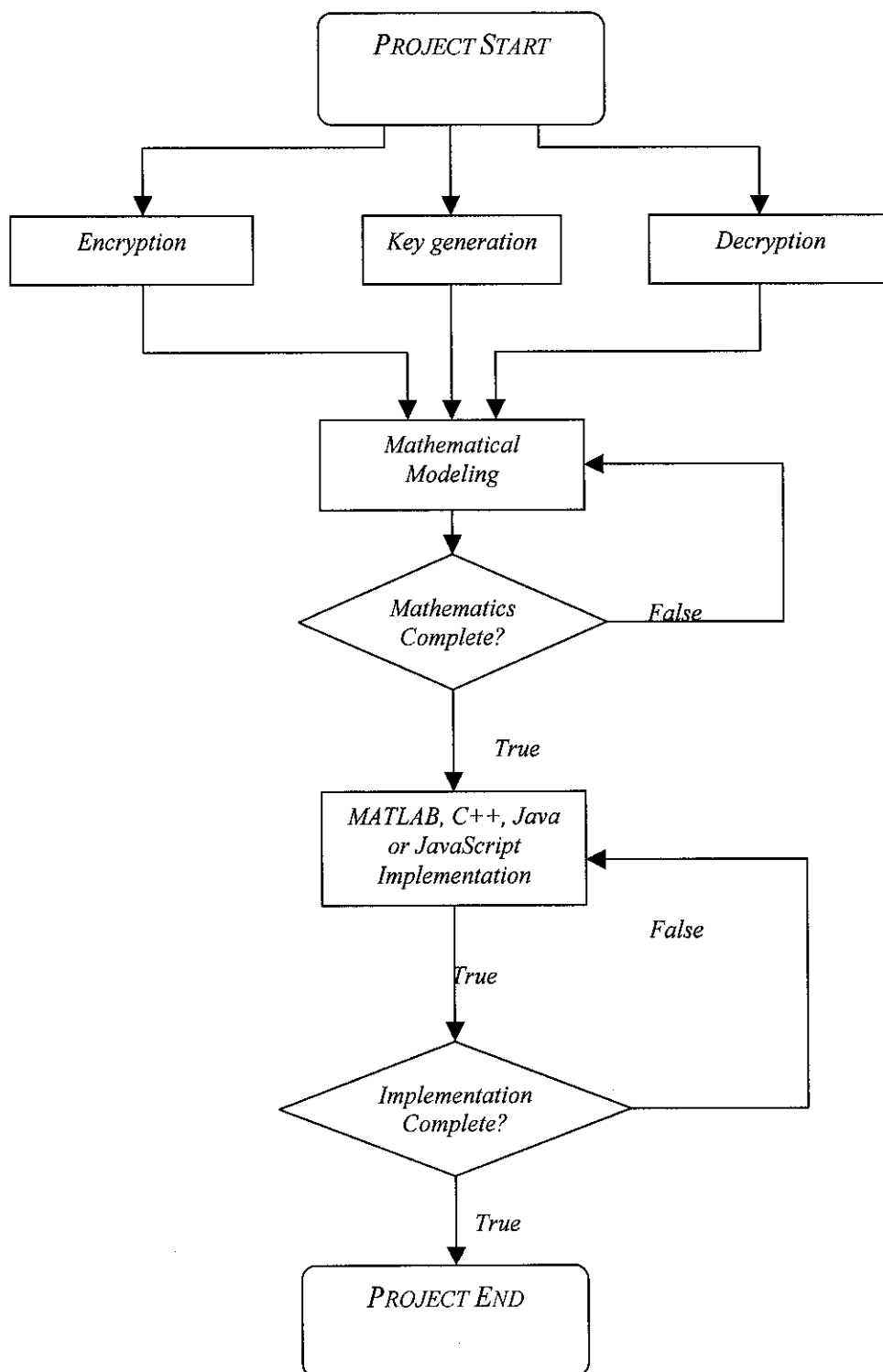


Figure 3.2: Project Flowchart

3.2 TOOLS REQUIRED

The system was supposed to test if the plaintext was still the same as before encryption and after encryption. It also supposed to test if the length of ciphertext is the as the length of the plaintext. Tools required achieving this task of designing a cipher need to be efficient and readily available. The software should enable measuring text length, and convert text characters into decimal numbers. Table 3.1 lists some of the equipment/tools that were envisaged to be necessary in the project.

Table 3.1: Lists of the equipment necessary for this project.

Project Equipment/Tools		
No.	Software	Hardware
1.	MATLAB	2*Computers
2.		Local Area Network

The cipher tested for this project is software based and has lot of advantages compared to the hardware based. The software-based cipher is cheaper because the user needs only the computer with the appropriate software, no need for extra hardware.

3.2.1 Software Based Cipher

MATLAB was chosen because it can create the Graphical User Interface (GUI) and implement diffusion and confusion. The MATLAB libraries are easy to use for debugging and provide a guide in effectively implementing different functions. Since all the libraries required for encryption come as standard package in MATLAB, it is convenient compared to Java, which requires addition libraries for advanced applications of encryptions. Any of these software; Java, C++ or JavaScript could be used and for mass distribution of the cipher. The use of C++ or Java has proven to be advantageous.

3.2.2 Public Network/Local Area Network

The plaintext is encrypted so that it can be stored or transmitted over the network in encrypted form. When the cryptosystem encrypts a text file, the file can be attached as document to an email or placed on sharing on the network or stored in the smartcard. The data is sent via email attachments or put on sharing needs communication public networks, like LAN.

In order to use insecure LAN, Microsoft operating system and the other operating systems provide the capability of folder sharing. This folder sharing allows the user to test if the data can be encrypted on one computer and decrypted on the other computer in the network.

The purpose of LAN was for transmission of encrypted data on the public network. The logic was to encrypt a file on one machine, transmit it over the network and decrypt it on the other computer. It is required that the cipher encrypts and decrypts on two different computers.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 RESULTS

4.1.1 Implementation of Chaotic Cipher

This projected implemented Logistic Map Cipher, which uses one key, hence symmetric. The implementation code for this cipher is given in appendix D. The cipher is strictly for plaintext encryption; other types of data can be encrypted if the algorithm is modified. The confusion and diffusion and iteration methods are used to avoid the redundancy that occurs in ciphertext. Redundancy is the tendency of ciphertext character repetition as the result of plaintext character repetition. In the MATLAB code developed for logistic map, diffusion and confusion are implemented with XOR function. The algorithms for most of the chaotic maps have iterations, diffusion and\ confusion as the main methods. These methods for chaotic maps are represented are explained below:

- **Key Breakdown**

The secret key, which is the initial condition for the chaotic map cipher is converted to a decimal number. This key is iterated for as long as there is characters of plaintext that need to be encrypted [4], [9]. Figure 4.1 below indicates how the key is broken down and used in iterations.

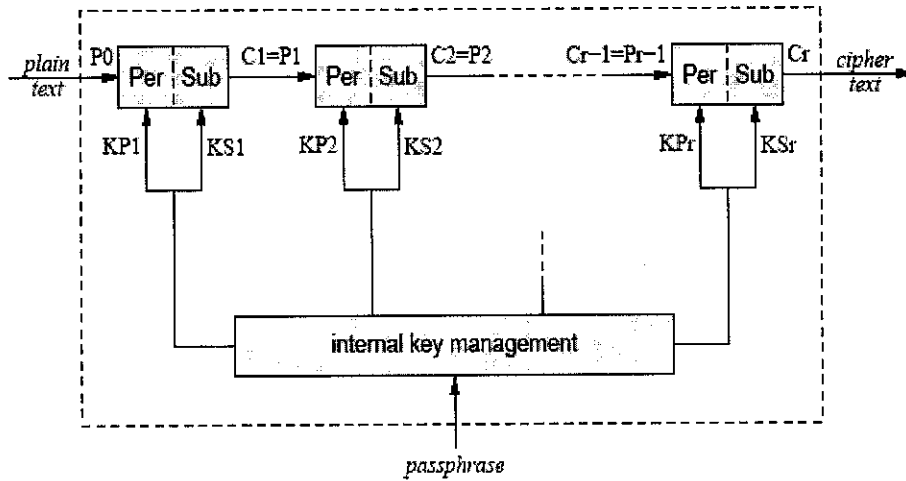


Figure 4.1: General block diagram for chaotic cipher internal mechanisms [10].

- **Iterations**

In chaotic encryption the plaintext is mixed with the secret key to generate the ciphertext. Because the resulting ciphertext sometimes has the traces of the plaintext, encryption takes place more than once. A block of cipher text is used as an input for second round of encryption; this process is repeated until the traces of the plaintext are removed in the cipher text. The number of iterations performed depends on the plaintext length chaotic cipher.

- **Confusion and Diffusion**

The symmetric and asymmetric keys are used more than once to mix with the plaintext so that the ciphertext can be obtained. The bits of ciphertext are removed from their original position to the new positions in the block of text. This makes it further difficult for cryptanalyst to trace the pattern of plaintext in ciphertext. This is for confusion purposes. Diffusion on the other hand replaces certain bits with new bits. The resulting ciphertext will contain no details or traces of the original plaintext. Further details on the two concepts are given in [4].

4.1.2 Logistic Map Ciphers Implementation

The encryption steps used for logistic cipher are iteration and XOR. The XOR function is used in place of diffusion and confusion steps, which are necessary to implement a chaotic cipher [10], [20]. In MATLAB the logistic equation does not iterate the plaintext characters as they are, the characters must first be converted into decimal number, double so that they can be XORed with the secret key x_0 of the logistic equation. Output of the logistic equation, which is the new value of x is multiplied with 10^7 so that it can be used to XOR it with plaintext. One example for the conditions for a symmetric cipher for encryption information in figure 4.3 is:

- $x_0 = 0.12345678978787879$
- $a = 3.999$

Figure 4.2 indicates a piece of code used to implement encryption results in figure 4.3. The *for* loop in line 10 of figure 4.2 is the one that implements encryption. The iteration numbers are as many as the length of the data to be encrypted. For example, the plaintext of two characters is achieved by 2 iterations.

```
3 mu=3.99;
4 k=input(' Enter the data \n ','s');
5 plain=double(k);
6 data_length=length(plain);
7
8 x0=0.1234567891;
9
10 for i=1:data_length
11
12     xn=mu*x0*(1-x0);
13     pik=floor(x0*(10^7));
14     pik=mod(pik,256);
15     ciphered(i)=bitxor(pik,plain(i));
16     x0=xn;
17 end
18
```

Figure 4.2: Encryption algorithm for logistic map

Steps 1 through 4 of figure 4.3 show the encrypted and decrypted data as normal characters or decimals.

1. Plaintext characters: *The data I am about to encrypt is important*

2. Plaintext in decimals:

Columns 1 through 15

84 104 101 32 100 97 116 97 32 73 32 97 109 32 97

Columns 16 through 30

98 111 117 116 32 116 111 32 101 110 99 114 121 112 116

Columns 31 through 43

32 105 115 32 105 109 112 111 114 116 97 110 116

3. Cipher text characters:

Ó2□â)q6PÔ7ñ^>L□CĚ¥Cù⊘DÈ||f«İn¼ "×O□İ"N□□9R

4. Cipher text in decimals:

Columns 1 through 15

211 50 22 229 41 113 54 80 210 55 241 94 62 76 143

Columns 16 through 30

67 203 165 67 249 164 208 200 124 166 102 171 204 110 188

Columns 31 through 43

160 168 215 79 152 108 237 34 78 147 147 57 82

Figure 4.3: Plaintext and Ciphertext in decimals and ASCII characters (Based on Figure 4.2)

Each character is represented as decimal with maximum of 3 digits. In MATLAB a character is placed in a column and is referenced by column number so that it can be used in the XOR process. In MATLAB the number of digits behind the decimal points for x_0 can be more than 64 digits and the speed of the cipher still remains noticeably the same. The security of the cipher increases with the number of digits behind the decimal points for the initial condition of x .

4.1.3 Analysis

The logistic equation, $y = a \cdot x_n \cdot (1 - x_n)$ [16] encrypts and decrypts only if the conditions (a and x_n) allow chaos to be achieved. Figure 4.4 indicates the values taken by the logistic equation at different iterations (n values). Every one of these values is used to encrypt one character in a plaintext. During encryption each character of plaintext is converted into 8 bits hexadecimal number. These 8 bits are encrypted as a block of bits, which makes this cipher a block cipher.

As figure 4.4 shows, the values of y are different iterations. The y values are on the vertical axis and the number of iteration on the horizontal axis. These y values will not saturate to one particular point, as long as the conditions for chaos of logistic equation were properly chosen.

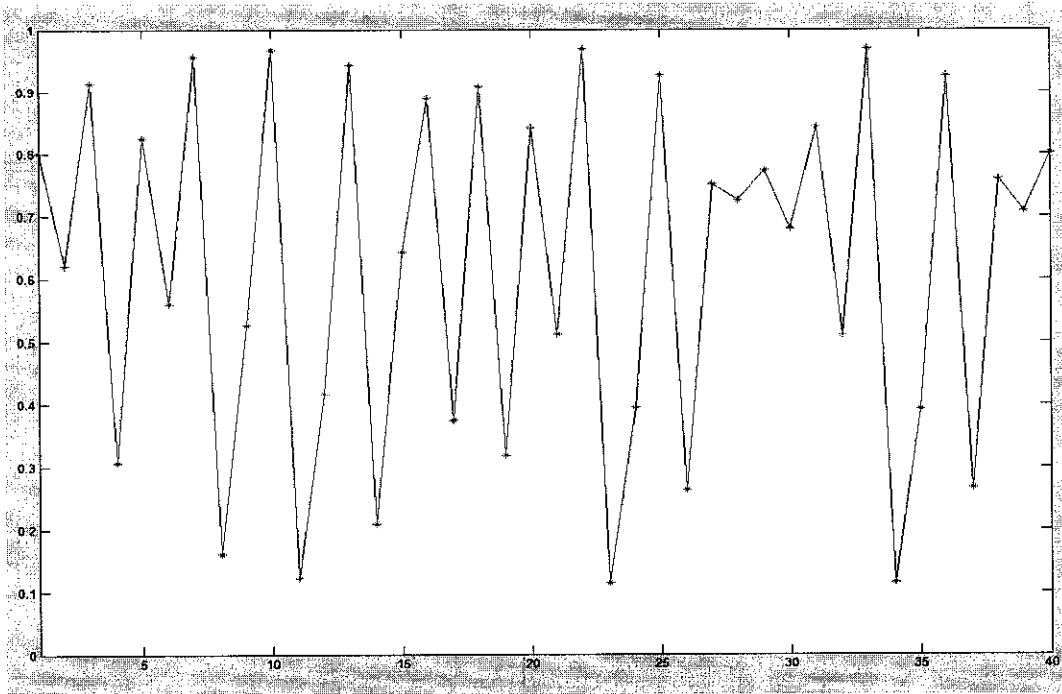


Figure 4.4: Iterated values for the logistic equation encrypting 40 characters (Based on Appendix B)

As the file size increases the number of iterations required increases and this may make this type of implementation of logistic map based cipher unfavourable for huge text files in computers with slow processing speed.

The resulting ciphertext is in ASCII [20] and each character represents one character of plaintext. If any of the ASCII characters (Appendix E) in the ciphertext is altered, the cipher does not return the plaintext during decryption. This prevents intruders from inserting their own text in the ciphertext. Since the plaintext length is the same as that of the ciphertext, the intruder can not add extra character in the ciphertext without the user realizing. The user only realizes when the cipher text does not show readable data,

4.1.4 Limitations

- i. The cipher is only limited to encryption of files that the user knows their extension. For example, if a file is pdf or txt type, the new file containing ciphertext should be saved with the name that ends with those extensions; *filename.pdf* or *filename.txt*. The person decrypting the file needs to be aware of the kind of extension the file was saved with so that the decrypted file can be saved with the proper extension format. For the files whose contents are ASCII characters, they may appear scrambled before encryption is done, hence the user encrypting needs to note that they were just plaintext before encryption.
- ii. The cipher does not encrypt files containing, pictures, graphics and voice attachments. If user tries to encrypt files with this kind of context, the program does not execute. Some modifications need to be made to the cipher proposed in this report so that it can be efficient and able to encrypt other types of data.
- iii. The secret key is give in the form a decimal, which it would be easy for intruders to guess using brute force attack. The preferred method of giving the keys is in hexadecimal, which can be stored in digital certificates. The hexadecimals of 128 bits [1], [15] are regarded long and safe enough to use as the secret keys for Logistic Map Cipher. And for the cipher developed in this project to be safe it would require that the decimal number for the key to be converted into a 128 bits hexadecimal key. The digital certificate would be used for the storage of such key.

4.2 DISCUSSIONS

The symmetric algorithm investigated for this project is based on Logistic Map. The attempt has been to understand how the algorithms for different encryption schemes work. The special attention paid to the chaotic algorithm was to test the possibility of using it for developing either symmetric or asymmetric cryptosystems. Symmetric ciphers offer better performance for bulk data encryption. The PKI system is suitable for cipher that encrypts few bytes of text. Hence for applications like video, voice and picture encryption it is better to use symmetric ciphers because they are faster.

The cryptanalysis of different cipher needs to be carried out robustly to avoid hackers breaking into the cryptosystem. The best ciphers can also be created from chaotic maps and other types for as long as the algorithm is not kept a secret. By making the algorithm known publicly it gives other fellow scholars and hackers the chance to try breaking into the cipher so that it can be modified if it is not secure.

Both hardware and software based cipher have advantages of their own. For gadgets like mobile phones and landline telephones, it is much easier to use hardware/firmware because they are firmware oriented. But with computer networks so dominant, it is favorable to implement software-based cipher. It may be because the computers readily use software.

The speed of converting text differs with the size of text being encrypted with the Logistic Map Cipher. Since the length of plaintext is the same as that of ciphertext it takes the same amount of time to encrypt plaintext and decrypt ciphertext. The cipher encrypts a whole block of bits that make up one ASCII character. The Logistic Map Cipher implemented in this safe pending cryptanalysis.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

The project investigated and reported on the use of different types of encryption systems used. It studied the encryption schemes that are established and those that are still being used for research purposes like, chaotic maps.

It has been found that the chaotic maps have been used in the past to develop both symmetric and asymmetric ciphers. They can still be used to develop secure encryption algorithms. The logistic map is even suitable for developing ciphers used for video and still pictures encryption.

The implementation of Logistic Map Cipher in this project used the simple method of iteration and XOR for encryption. Decryption uses the same process as encryption; the difference is that the input for the cipher is the ciphertext instead of plaintext.

5.2 RECOMMENDATIONS

It is recommended that the Logistic Map Cipher implemented in this project be subjected to cryptanalysis to ascertain its level of security before further modifications can be made. The encryption of files should be made user friendly so that it does not require the user to know the extension of the file when encrypting or decrypting. Hence a proper interface for the program still needs to be designed.

REFERENCES

- [1] Steve Burnett and Stephen Paine, 2001, RSA Security's Official Guide to Cryptography, USA, McGraw-Hill Press.
- [2] Nigel Smart, 2003, Cryptography: An Introduction, UK, McGraw-Hill Publication.
- [3] Jalal Feghhi, Jalil Feghhi, Peter Williams, 1998, Digital Certificates: *Applied Internet Security*, USA, Adison Wesley Publication.
- [4] David Hook, 2005, Beginning Cryptography with Java, USA, Wiley Publishing Inc.
- [5] John R. Vacca , 2004. Public Key Infrastructure: *Building Trusted Applications and Web Services*, USA, Auerbach Publications.
- [6] Nick Galbreath, 2002, Cryptography for Internet and Database Applications: *Developing Secret and Public Key Technique with Java*, Canada, Wiley Publishing Inc.
- [7] Man Y Rhee, 2003, Internet Security: *Cryptographic Principles, Algorithms and Protocol*, England, John Wiley & Sons Ltd.
- [8] Bruce Schneier, 1996, Second Edition, Applied Cryptography: *Protocols, Algorithms, and Source Code in C*, Canada, John Wiley & Sons, Inc.
- [9] Michael P Kennedy, Riccardo Rovatti and Gianluca Setti, 2000, Chaotic Electronics in Telecommunications, USA, CRC Press LLC.
- [10] Lorenzo Cappelletti, 2000, An FPGA Implementation of Chaotic Encryption Algorithm, University of Padova,
- [11] Pina Bergamo, Paolo D'Arco, Alfredo De Santis, Ljupco Kocarev, November 2004, Security of Public Key Cryptosystem based on Chebyshev Polynomials, arXiv:cs/0411030.
- [12] Roy Tenny and Lev S. Tsimring, January, 2004, Steps towards Cryptanalysis of Chaotic Active/passive Decomposition Encryption Schemes Using Average Dynamics Estimation, International Journal of Bifurcation and Chaos, Vol. 14, No. 11 (2004), 3949 – 3968.
- [13] Po-Han Lee, Soo-Chang Pei and Yih-Yuh Chen, December, 2003, Chinese Journal of Physics, Vol.41, No.6, 559 – 581.

- [14] R. A. McNees, V. Protopopescu, R.T. Santoro, J. S. Tolliver, August 1993, Cryptosystem based on Chaotic Dynamics, Oak Ridge National laboratory, Central Research Library, Circulation Section 4500N, Room 175.
- [15] S. Papadimitriou, T. Bountis, S. Mavroudi, and S. Bezerianos, 2001, A Probabilistic Symmetric Encryption Scheme for Very Fast Secure Communication Based on Chaotic Systems of Difference Equations, International Journal of Bifurcation and Chaos, Vol. 0, No. 0, 000 – 000.
- [16] Logistic Map on Wikipedia, URL: http://en.wikipedia.org/wiki/Logistic_map, 01 – 06 – 2006.
- [17] A. Menezes, P. van Oorschot, and S. Vanstone, 1996, Handbook of Applied Cryptography, USA, CRC Press, Inc, 283 – 312.
- [18] Kwok-wo Wong, Kwan-Pok Man, Shujun Li and Xiaofeng Liao, Sep – Oct, 2005, A More Secure Chaotic Cryptographic Scheme based on Dynamic Lookup Table, Circuits Systems and signal Processing 24 (5), 571 – 584.
- [19] C. Koc and T. Acar, May 1997, A Methodology for High-Speed Software Implementations of Number-Theoretic Cryptosystems, Oregon State University, Technical Paper.
- [20] S. Midorikawa, T. Kubo, T Cheon, 28 Dec 1995, Folded Bifurcation in Coupled Asymmetric Logistic Maps. arXiv:chao-dyn/9512007, v1.
- [21] Symmetric Encryption Modes or Opearation, URL: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation, 01 – 06 – 2006.
- [22] The Tiny Encryption Algorithm graphical user interface, URL: <http://www.codeproject.com/netcf/teaencryption.asp?df=100&forumid=33635&exp=0&select=1047427>, 19 – 06 – 2006.
- [23] MATLAB source codes for logistic map equation, URL: <http://www.humboldt.edu/~cd43/561f05/logistic/logistic2.m>, 19 – 06 – 2006.

APPENDICES

Appendix –A: Tools and Materials Required

The following is a list of initial materials and tools necessary to complete the project.
This data will be provided at a later stage, after design calculations.

- a. MATLAB software.
- b. A personal computer that can support the software in (a).
- c. One file, to test encryption and decryption

Table A: Logistic Equation Parameters

Logistic Equation’s Parameters	Values
Chaos constant, r	[3.5 to 4]
Initial condition, x	(0 to 1)
The Iterations, n	Plaintext length

Appendix -B: Source Codes Iterations Projection (MATLAB) [23]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% COMPUTING TRAJECTORIES OF THE LOGISTIC  
MAP %%%%%%%%%%
```

```
a=3.88      % Parameter value
```

```
x0=0.8      % Initial condition
```

```
N=40;      % Number of iterations
```

```
x(1) = x0;   % Matrices in matlab cannot have zero index
```

```
% compute the orbit and print out results
```

```
fprintf(1,'Year    Density\n');
```

```
for ic=1:N
```

```
    x(ic+1) = a*x(ic)*(1-x(ic));
```

```
    fprintf(1,'%d    %f\n',ic,x(ic+1));
```

```
end
```

```
% graph the orbit
```

```
figure(1); hold off;
```

```
plot(x,'r*');
```

```
hold on;
```

```
plot(x,'b');
```

```
axis([1 N 0 1]);
```

Appendix –C: Source Codes Bifurcation Diagram (MATLAB) [23]

Logistic map bifurcation diagram and time series plot

% Demonstration for course b8: Nonlinear Systems

% M.Little, 2005

% Post-transient map iteration length

N = 100;

% Transient removal length

T = 100;

% Number of points on the bifurcation parameter axis to evaluate

M = 500;

% Initial condition

y0 = 0.7;

% Minimum and maximum lambda values

lambda_min = 0;

lambda_max = 4;

% Loop initialisation

lambda_delta = (lambda_max - lambda_min)/M;

lambda = lambda_min;

% Bifurcation parameter scan loop

for m = 1:M

 % Logistic map transient iteration

 y = y0;


```

for n = 1:T
    y = lambda*y*(1-y);
end

% Logistic map iteration
bif_y(1,m) = y;
bif_x(1,m) = lambda;
for n = 1:N
    bif_y(n+1,m) = lambda*bif_y(n,m)*(1-bif_y(n,m));
    bif_x(n+1,m) = lambda;
end

% Update map parameter
lambda = lambda + lambda_delta;
end

% Plot bifurcation diagram
figure;
plot(bif_x, bif_y, 'k.', 'MarkerSize', 1);
title('Logistic map bifurcation diagram');
xlabel('\lambda');
ylabel('y');

```

Appendix –D: Implemented Source Code for Logistic Map Cipher (MATLAB)

```
%This is the Logistic Map cipher that was developed for this project, A Study and  
%Implementation of Encryption, with Emphasis on Chaotic Maps  
%Reads the file into a cipher
```

```
[fn, pn] = uigetfile('*.txt', 'Plis Specify the Filename');
```

```
k = fopen(fn, 'rt+');
```

```
A = fread(k);
```

```
disp(char(A')); % converts the content of the file into plaincharacter for
```

```
    % MATLAB manipulation
```

```
% The beginning of the code for encrypting plain text
```

```
mu=3.999;          % The chaos constant (3.5 to 4)
```

```
plain=double(A);    % Convert file contents into decimals,
```

```
data_length=length(plain); %Fetches the length of the data to be encrypted
```

```
x0=0.123456789;    % The secret key for the logistic map (0, 1)
```

```
%Beginning of the iteration for loop
```

```
for i=1:data_length
```

```
    xn=mu*x0*(1-x0);    % Logistic equation
```

```
    multi=floor(x0*(10^7));
```

```
    multi=mod(multi,256); % Converts the plaintext into ASCII characters
```

```
    ciphered(i)=bitxor(multi,plain(i)); % XOR plaintext with the secret key
```

```
    x0=xn;
```

```
end
```

```
%Saving the workspace of ciphertext
```

```
[fn1, pn1] = uiputfile('*.txt', 'Save workspace as');
```

```
k1 = fopen(fn1, 'wt');
```

```
%fwrite(k1, deciphered, 'char');
```

```

fwrite(k1, ciphered, 'char');
status = fclose('all');

%Beginning of the code for decrypting the cipher text
x0=0.123456789;

for i=1:data_length

    xn=mu*x0*(1-x0);
    multi=floor(x0*(10^7));
    multi=mod(multi,256);
    deciphered(i)=bitxor(multi,ciphered(i));
    x0=xn;
end

%Saving the workspace of ciphertext
[fn1, pn1] = uiputfile('*.mat', 'Save workspace as');
k1 = fopen(fn1, 'wt');
fwrite(k1, deciphered, 'char');
status = fclose('all');

%End the the programme

```

Appendix –E: Representation of ASCII characters

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.asciitable.com

Appendix E (continues)

128	Ç	144	É	160	á	176	☒	193	⊥	209	⚏	225	ß	241	±
129	ü	145	æ	161	í	177	☑	194	⌊	210	⌋	226	Γ	242	≥
130	é	146	Æ	162	ó	178	☐	195	⌈	211	⌌	227	π	243	≤
131	â	147	ø	163	û	179		196	—	212	⌍	228	Σ	244	∫
132	ä	148	ö	164	ñ	180	⌋	197	+	213	⌎	229	σ	245	∫
133	à	149	ò	165	Ñ	181	⌌	198	⌋	214	⌏	230	μ	246	+
134	ã	150	û	166	²	182	⌍	199	⌌	215	⌐	231	τ	247	≈
135	ç	151	ù	167	°	183	⌎	200	⌍	216	⌑	232	Φ	248	°
136	ê	152	—	168	¿	184	⌏	201	⌎	217	⌒	233	⊖	249	.
137	ë	153	Ö	169	—	185	⌐	202	⌏	218	⌓	234	Ω	250	.
138	è	154	Ü	170	¬	186	⌑	203	⌐	219	■	235	δ	251	√
139	í	156	£	171	½	187	⌒	204	⌑	220	■	236	∞	252	—
140	î	157	¥	172	¾	188	⌓	205	=	221	▬	237	φ	253	²
141	ï	158	—	173	¡	189	⌔	206	⌐	222	▬	238	ε	254	■
142	Ä	159	f	174	«	190	⌕	207	≠	223	■	239	∧	255	
143	Å	192	Ł	175	»	191	⌖	208	⌕	224	⌘	240	≡		

Source: www.asciitable.com

ASCII Code Table (Extended List)

Appendix –F: Source Codes for Symmetric Cipher with Padding [4] (Java)

```
//Symmetric Cipher;
/**
 *Basic symmetric encryption example
 */
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class SimpleSymmetricPaddingExample
{

    public static void main(String[] args)throws Exception

    {byte[] input = new byte[] {
        0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
        0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
        0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17};

    byte[] keyBytes = new byte[]{
        0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
        0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
        0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17};

    SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");

    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding", "BC");

    System.out.println("input text : " + input);//Utils.toHex(input));

    //encryption pass
```

```

cipher.init(Cipher.ENCRYPT_MODE, key);

byte[] cipherText = new byte[cipher.getOutputSize(input.length)];

int ctLength = cipher.update(input, 0, input.length, cipherText, 0);

ctLength += cipher.doFinal(cipherText, ctLength);

System.out.println("cipher text: " + Utils.toHex(cipherText) + " bytes: " +
ctLength);

// decryption pass

cipher.init(Cipher.DECRYPT_MODE, key);

byte[] plainText = new byte[cipher.getOutputSize(ctLength)];

int ptLength = cipher.update(cipherText, 0, ctLength, plainText, 0);

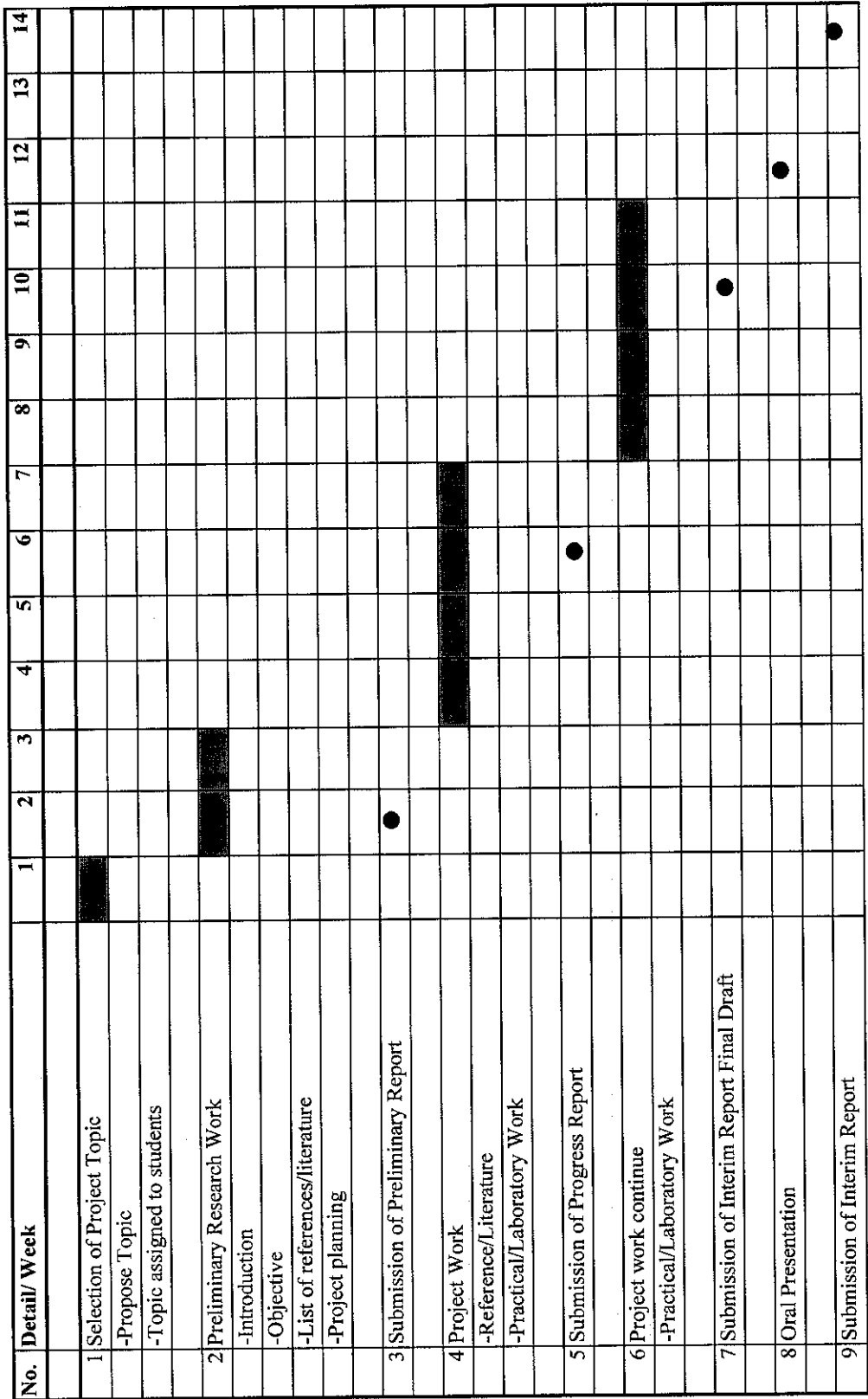
ptLength += cipher.doFinal(plainText, ptLength);

System.out.println("plain text : " + Utils.toHex(plainText) + " bytes: " + ptLength);

//end of program
}
}

```

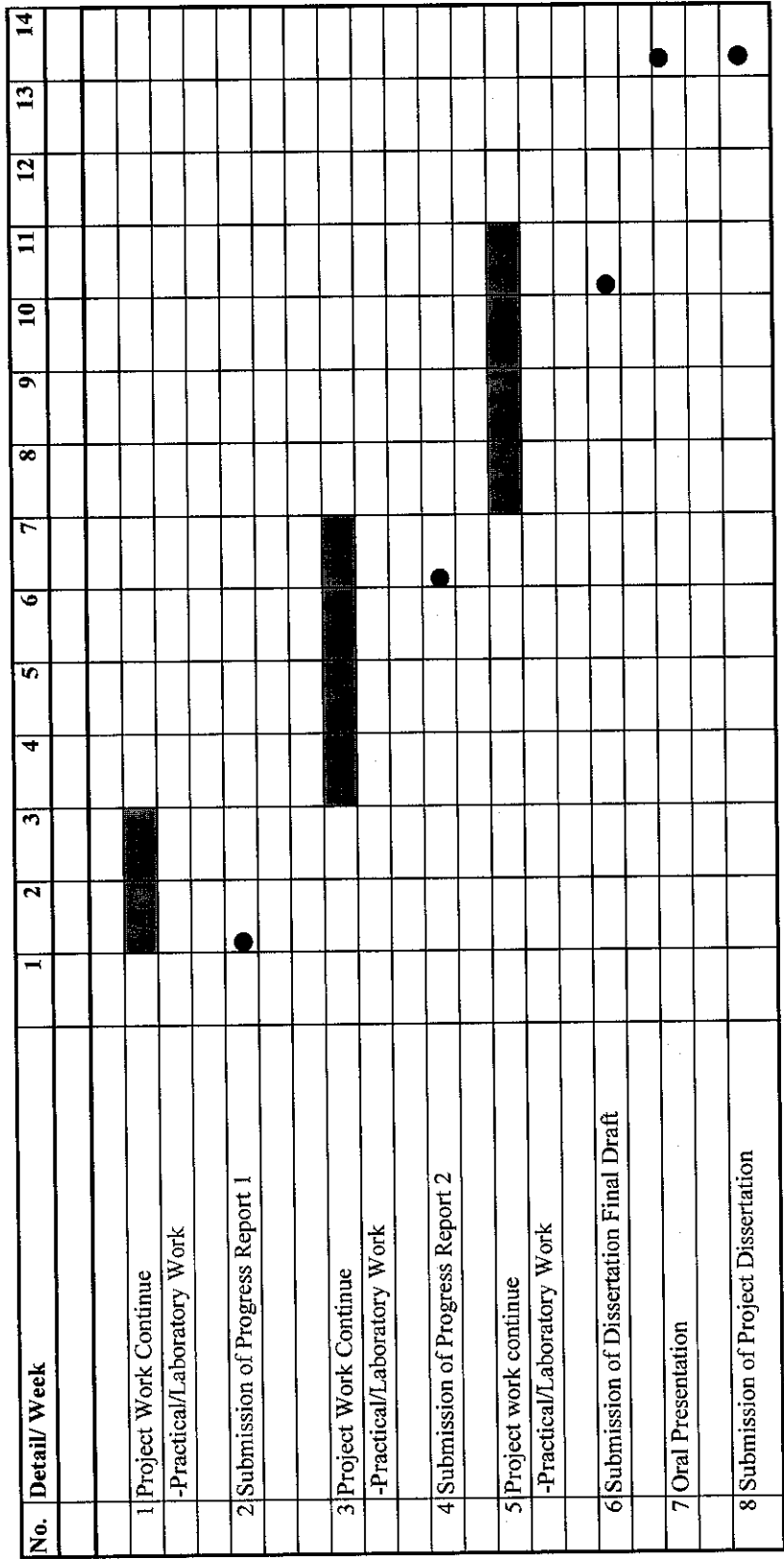
Appendix –G: Project Gantt chart
First Semester of Final Year Project



● Report submissions
 ■ Progress

Appendix –G: Project Gantt chart

Second Semester of Final Year Project



● Report submissions
■ Progress